RATER NOTIFICATION SYSTEM FOR COMPUTERIZED ASSISTED SCREENING TOOL

_____

A Thesis

Presented to the

Faculty of

San Diego State University

_____

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

_____

by

Pooja Jaymitkumar Shah

Fall 2011

SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Thesis of Pooja Jaymitkumar Shah:

Rater Notification System for Computerized Assisted Screening Tool

Carl Eckberg, Chair
Department of Computer Science

William Root
Department of Computer Science

Mary Ann Lyman-Hager
Language Acquisition Resource Center

Oct 10, 2011
Approval Date

Copyright © 2011
by
Pooja Jaymitkumar Shah
All Rights Reserved

DEDICATION

This thesis is dedicated to my parents Jaymitkumar Shah and Hina Shah, and my brother Rushi who always have been a source of encouragement and inspiration to me.

ABSTRACT OF THE THESIS

Rater Notification System for Computerized Assisted Screening
Tool
by
Pooja Jaymitkumar Shah
Master of Science in Computer Science
San Diego State University, 2011

Computer Assisted Screening Tool (CAST) gauges oral proficiency of students in various languages. It allows test takers to take an on-line oral interview consisting of automated audio and visual elicitation prompts. Once the test has been taken, it is evaluated by a person who is an expert in the language in which the test has been taken. After evaluation test takers receive results indicating whether or not they achieved advance level with feedback on which areas need to be improved.

The Rater Notification System for CAST enhances the functionalities of the existing rating process. This system sends notification by email to the raters when the test taker completes his test. It implements a locking/unlocking mechanism to prevent multiple raters from evaluating the same test at the same time, by restricting the rating process to only one rater and unlocks it on different circumstances like browser closing, session timeout, user navigation to a different page and logout. It provides a timer to count the time for the rating process and notifies the rater when the timer is about to expire, with an option to extend the timer.

TABLE OF CONTENTS

PAGE

ABSTRACT v
LIST OF TABLES viii
LIST OF FIGURES ix
LIST OF ACRONYMS x
ACKNOWLEDGEMENTS xi
CHAPTER
1 INTRODUCTION 1
1.1 Computerized Assisted Screening Tool (CAST) 1
1.2 The Existing Rating Process of CAST 2
1.3 Purpose of Rater Notification System 2
2 LANGUAGE ACQUISITION RESOURCE CENTER 3
3 TECHNOLOGIES USED 5
3.1 Servlets 5
3.2 Oracle 6
4 PROBLEM DESCRIPTION AND PROPOSED SOLUTION 8
5 DESIGN AND IMPLEMENTATION 10
5.1 Development Environment for CAST 10
5.2 Email Notification to Raters 10
5.3 Locking the Rating Process 12
5.4 Implementation of Timer 15
5.5 Unlocking 18
5.6 Issues with Locking and Unlocking 23
6 SECURITY 26
6.1 SQL Injection Overview 26
6.2 SQL Injection Prevention Techniques 27
6.3 Techniques Used in This Project 28
7 FUTURE ENHANCEMENTS 29

REFERENCES 30
APPENDIX
USER MANUAL.....31

LIST OF TABLES

PAGE

Table 5.1. Table Structure for “ADMINS” Table.....11
Table 5.2. Table Structure for “LANGS” Table.....12
Table 5.3. Table Structure for “BETAREG” Table.....14

LIST OF FIGURES

PAGE

Figure 5.1. Flowchart depicting the steps involved in sending emails. 13
Figure 5.2. Email sent on test completion.....14
Figure 5.3. List of tests viewed by the rater when he clicks on ‘Rate CAST Test.’.....15
Figure 5.4. Rating page for testid ‘XapMNo.’.....16
Figure 5.5. Alert message when another rater tries to rate the same test.....16
Figure 5.6. Flowchart depicting the steps involved in executing the timer. 19
Figure 5.7. The custom alert message to notify the rater about the timer.....20
Figure 5.8. Flowchart depicting the locking and unlocking process. 22
Figure 5.9. A rater trying to rate the same test on another tab.....24
Figure 5.10. Alert displayed when a rater tries to rate the same test on another tab. 24
Figure 5.11. Warning on Chrome when rating starts.....25

LIST OF ACRONYMS

ACTFL American Council on the Teaching of Foreign Languages

API Application Programming Interface

ASP.NET Active Server Pages .NET

CAST Computerized Assisted Screening Tool

CGI Common Gateway Interface

DBA Database Administrator

DBMS Database Management System

HTML Hyper Text Markup Language

HTTP Hyper Text Transfer Protocol

JDBC Java Database Connectivity

JSP Java Server Pages

LARC Language Acquisition Resource Center

MIME Multipurpose Internet Mail Extensions

OPI Oral Proficiency Interview

OS Operating System

PHP Hypertext Preprocessor

PL/SQL Procedural Language/Structured Query Language

RAM Random Access Memory

RDBMS Relational Database Management System

SDK Software Development Kit

SQL Structured Query Language

URL Uniform Resource Locator

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Prof. Carl Eckberg for his support, kindness and guidance. I would like to thank the rest of my thesis committee: Prof. Mary Ann Lyman-Hager and Prof. William Root for their encouragement and insightful comments. My sincere thanks also go to Michael Pasamonik for his assistance and support. Last but not the least, I would like to thank my family: my parents Jaymitkumar Shah and Hina Shah, and my brother Rushi for their continuous support and interest in what I do.

CHAPTER 1

INTRODUCTION

1.1 COMPUTERIZED ASSISTED SCREENING TOOL (CAST)

CAST (Computer Assisted Screening Tool) is designed to gauge oral proficiency of students in various languages - English, Spanish, French, Modern Standard Arabics, Persian, Egyptian Arabics, Mandarin Chinese, Iraqi Arabic and Hindi [1]. A consortium of five educational and professional institutions, San Diego State University, Brigham Young University, the Defense Language Institute Foreign Language Center, the Center for Applied Linguistics, and the American Council on the Teaching of Foreign Languages has collaborated in an effort to develop a pre-ACTFL OPI screening test. This test allows test takers an opportunity to take an on-line oral interview consisting of automated audio and visual elicitation prompts that are used to provide informative feedback regarding their oral ability. CAST exists only at the advance level. The main function of CAST is to screen whether or not the candidates' language proficiency has reached the advance level.

1. Features of CAST:
    - The CAST is meant to check the oral proficiency of a student in the target language.
    - The student can register for the test only after the administrator approves it.
    - The results with feedback comments are provided by email.
    - The test is evaluated by the language expert in the target language.

In order to appear for a CAST test, a test taker has to first register for the test. The registration details go to the administrator for approval. The administrator checks and approves the test taker's registration and then sends the test taker the login information for the test which includes testid for the test. The test taker can begin the test by logging in with his testid and email. Once the test has been taken, it is evaluated by a person who is expert in the language in which the test has been taken. After evaluation, test takers not only receive results indicating whether or not they achieved advance level but also receive feedback on which areas need to be improved and how their performance can be improved.

1.2 THE EXISTING RATING PROCESS OF CAST

This section describes the rating process, before the application in this thesis was implemented. The rating process allows the rater, a person who evaluates the test, to login and evaluate a test submitted by the test taker. Each rater is a language expert and is authorized to evaluate the tests taken in this particular language. The rater can view all the unrated tests in his registered language. After completing the evaluation, the rater can send evaluation results with the feedback comments to the test taker. The rater has to manually find out whether there are any tests ready for evaluation by logging into his CAST account. When a test taker finishes the test there is no way to notify the target language raters about the submitted tests. Once the evaluation process is started, the rater can take as much time as he wants to finish the rating process. There is no timed restriction or notification for the rating process. Also, two raters can simultaneously evaluate the same test. But as only rater can submit the evaluation result, the other rater will be notified only when he will try to submit his results. Thus code developed in this thesis improves upon the process detailed above.

1.3 PURPOSE OF RATER NOTIFICATION SYSTEM

The Rater Notification System is designed to focus on the above problems. It sends email notification to the mail accounts chosen by the raters to inform them about the test completion. To restrict the rating process to only one rater, it uses a locking mechanism by which it locks a test when the rater starts evaluation. It also unlocks the test on some predefined circumstances (browser closing, session timeout, navigation, logout). It uses a timer to count the time for the rating process and notifies the rater when the timer is about to expire, with an option to extend the timer. It unlocks the test when the timer expires. This solves the problem of two different raters working on one test at the same time.

A user manual for the Rater Notification System will be provided to the raters (see Appendix).

CHAPTER 2

LANGUAGE ACQUISITION RESOURCE CENTER

Language Acquisition Resource Center, LARC, is a national Language Resource Center located on the San Diego State University campus. Parts of this chapter are quoted from this website [2]. LARC develops and supports the teaching and learning of foreign languages in the United States through research, technology, and publications. LARC mainly pays attention to less commonly taught languages, cross-cultural issues, language skills assessment, and teacher training. LARC is sponsored by the San Diego State University Research Foundation (SDSURF) and is one of fifteen Department of Education Title VI funded Language Resource Centers (LRCs). The Language Resource Centers (LRCs) were established by Congress in 1989. Their purpose is to improve the teaching and learning of foreign and second languages in the United States. Summer Institutes are a major part of LARC's role as a national Foreign Language Resource Center in accordance with U.S. Department of Education Title VI grant funding.

LARC provides Multimedia Computer Labs facility to SDSU world language students and faculty to give rich, engaging media to support and enhance language learning. LARC labs offer an extensive set of materials and facilities, extended open hours, one-on-one assistance and a comfortable environment to support the acquisition and study of language.

LARC is collaborating with the University of Arizona's LRC in co-sponsorship of two Intercultural Competence Symposia in 2012 and 2014. LARC is also working on developing a Gaming Project. The goal of this project is to develop an approach in establishing a literacy and standards framework for pedagogical adaptations to games as language learning/literacy acquisition strategies.

For testing and assessment in different languages, LARC uses the Computerized Assisted Screening Tool (CAST). CAST can be effectively used by instructors as a tool to rate their students' oral skill level.

LARC hosts an annual free Social Media Workshop in a blended live and online format. The workshop provides real classroom examples and hands-on practice, through a series of language educators demonstrating their classroom uses of social media, language learning objectives, instruction on how to use the tool and also an activity that allowed participants to sign up and experiment with the new tool.

CHAPTER 3

TECHNOLOGIES USED

CAST is developed using Servlets and Oracle 10g database. This section briefly discusses these technologies.

3.1 SERVLETS

A servlet is a Java programming language class used to extend the capabilities of servers that host applications accessed via a request-response programming model [3]. A servlet can be thought of as an applet that runs on the server side. Java Servlet technology provides Web developers with a simple, consistent mechanism for extending the functionality of a Web server and for accessing existing business systems. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by Web servers. For such applications, Java Servlet technology defines HTTP-specific servlet classes [4]. A Servlet is a Java class in Java EE that conforms to the Java Servlet API, a protocol by which a Java class may respond to requests. Servlets are the Java counterpart to non-Java dynamic Web content technologies such as Common Gateway Interface (CGI), ASP.NET, and PHP.

Servlets can maintain state in session variables across many server transactions by using HTTP cookies, or URL rewriting. To deploy and run a Servlet, a Web container must be used. A Web container is essentially the component of a Web server that interacts with the servlets. The Web container is responsible for managing the lifecycle of servlets, mapping a URL to a particular servlet and ensuring that the URL requester has the correct access rights. Servlets can be generated automatically from Java Server Pages (JSP) by the Java Server Pages compiler. The difference between Servlets and JSP is that Servlets typically embed HTML inside Java code, while JSPs embed Java code in HTML. Three methods are central to the life cycle of a servlet. These are “init ( )”, “service ( )”, and “destroy ( )”. They are implemented by every servlet and are invoked at specific times by the server.

Earlier, CGI was developed to provide server side capabilities to web applications. Using servlets, web developers can create fast and efficient server side applications and can run them on any servlet enabled web server. A servlet runs entirely inside the Java Virtual Machine. Since the servlet runs on the server side, it does not depend on browser compatibility. Servlets have a number of advantages over CGI and other API's. They are:

- Platform Independence
- Performance
- Extensibility
- Safety
- Security

3.2 ORACLE

Oracle Databases are object-relational database management systems produced and marketed by Oracle Corporation [5]. This thesis has used only relational features, and thus is an RDBMS. The Oracle RDBMS stores data logically in the form of tablespaces and physically in the form of data files. Oracle's PL/SQL, a procedural language extension to SQL, is commonly used to implement program logic modules for applications. PL/SQL can be used to build stored procedures and triggers, looping controls, conditional statements, and error handling. PL/SQL procedures can be compiled and stored in the database.

Oracle has many advantages and features that make it popular. Oracle releases come with new features implemented in the new version so that the features of earlier versions are still maintained. Thus Oracle databases tend to be backwards compatible. Oracle is a reliable and adequate database system and its added features pass the ACID test **(**Atomicity, Consistency, Isolation, Durability**)**, which is important in insuring the integrity of data [6].

Oracle Database 10g, released in 2003, enables grid computing. A grid is simply a pool of computers that provides needed resources for applications on an as-needed basis. The goal is to provide computing resources that transparently scale to the user community. One important key to Oracle Database 10g's usefulness in grid computing is the ability to provision CPUs and data. Oracle Database 10g further reduces the time, cost, and complexity of database management through the introduction of self-managing features such as the Automated Database Diagnostic Monitor, Automated Shared Memory Tuning, Automated Storage Management, and Automated Disk Based Backup and Recovery.

One of the main advantages of oracle over other databases is in its recent version oracle has the concept of Flashback technology. Flash technology takes care of the situation where deletion of valuable data, deletion of the wrong data, or dropping the wrong table occur because of the hardware failure or human errors. By Flash technology it helps in recovery by working just on the changed data. Thus Flashback provides an efficient recovery from human errors, faster database recovery and helps in simplifying the management and administration processes.

The original CAST system was written using Servlets and Oracle, so Servlets and Oracle were really a system requirement for this thesis.

CHAPTER 4

PROBLEM DESCRIPTION AND PROPOSED SOLUTION

This section briefly discusses the problems in the existing rating process of CAST. There are many features in the CAST rating process which require enhancement. The Rater Notification System mainly focuses on the following problems:

1. The first problem is the inability of raters to find out about unrated tests automatically. When a test taker completes his test, the test remains in the system as an unrated test until it is evaluated. Only the rater who is a language expert in the test taker's language can rate the test and must do so by logging into his CAST account. When rater logs in, he can see 'Rate CAST Test' link on his home screen. Clicking on this link will display all the unrated tests in the rater's registered language. The only way for the rater to know about a new test submission is by logging into his account. There is no way to let the raters know about unrated tests automatically. If the raters do not check their accounts frequently, it may happen that the students have to wait for a longer time to get the test results.

   Proposed Solution: The solution to this problem is to let the raters know when the test is completed. The raters will be notified automatically on test completion. When a test taker completes a test, an email will be sent to all the raters of the target language about the new submitted test. The email will provide the information about the test (testid of the test) and a link to navigate to the rater's home screen.

2. Second problem is the ability of multiple raters to rate the same test at the same time under the previous system. The rating page of a specific test is not restricted to be viewed only by one rater at a time. When a rater starts rating a test, other raters cannot know about it. For other raters, the test is still there in the list of unrated tests and hence any rater can start rating the same test any time, which is already in the evaluation process by other rater. There isn't any mechanism which can prevent other raters from rating the same test and which can notify them that the test is already under review by another rater. Also, the list of unrated tests is not getting updated with the current unrated tests. The list should not show the tests which are currently under the rating process.

   Proposed Solution: The solution is to use a locking mechanism on the rating process. So that when a rater starts rating, no other rater can rate the same test at the same time. The test will be locked when the rating starts. When more than one rater will try to rate the same

test, the first of them will be able to start rating but the subsequent raters will get an alert message stating that the test is already being rated by another rater. The test will then disappear from the list of unrated tests. The list will be updated when the page is reloaded or refreshed. The test will be unlocked if the rater fails to finish the evaluation. The unlocked test will again appear in the list of unrated tests and it can be rated by any other rater of that specific language. There are many scenarios to be considered for locking/unlocking a test: The test will be locked when:

- A rater starts rating

The test will be unlocked, and appear in the unrated test list, when:

- There is a session timeout
- Rater logs out without submitting the results of rating
- Rater navigates away from the rating process
- Rater closes the tab or browser window
- Rater clicks on back button
- Rater does not complete rating in the specified time period

3. Third problem is there is no limit on the amount of time allowed for the rating process. The rater can take as much time as he wants to finish the rating. There isn't any specific time frame in which the rater is expected to finish rating. Also, the rater isn't being notified about the amount of time he has already spent on rating a particular test. If a rater has started rating and he is doing some other work in parallel, he might keep the rating page open for a longer time. By the time the rater finishes his rating, it might happen that another rater finishes rating the same test before him.

Proposed Solution: The solution is to allocate some predefined amount of time to the rating process. If the rater does not finish rating in the allocated time, he will be prompted to extend the time. When a rater starts rating, a timer will start incrementing. The rater will be prompted about the remaining time when the timer will reach to some predefined limit. The rater can extend the time limit by clicking on the prompt. The timer will then be reset and will again prompt the rater after a predefined limit. If the rater does not click on the prompt, the timer will expire after sometime and it will automatically navigate away from rating page. The test will then be unlocked and the test will appear in the list of unrated tests.

Clearly the timer concept became necessary when locking was added, so that obtaining a lock and getting distracted for a long interval, would not cause long delays in the rating process.

CHAPTER 5

DESIGN AND IMPLEMENTATION

This section describes the design and implementation of the proposed solution discussed in the previous section and issues that were faced. The solution works for Internet Explorer, Firefox, Safari, and Chrome browsers.

5.1 DEVELOPMENT ENVIRONMENT FOR CAST

The details of the development environment for CAST are given below:

1. Hardware Environment
    - Xserve PowerPC G5 (Type: 64Bit OS and Speed: 2.0 Ghz)
    - 2 GB RAM
    - 153 GB Hard Disk
2. Software Environment
    - Mac OS X Serve v 10.4.11
    - Java Standard Edition v 1.5 SDK
    - Orion
    - Oracle 10g Database

CAST is developed using Java Servlets as business logic, JavaScript and HTML as front-end, and Oracle database as back-end. The servlets for the rating process already exist in the CAST system. So, the major part of my project involved modifying already existing servlets. The servlet “rate.java” performs all the functionality of the rating process, including displaying the list of unrated tests, and submitting the rating results.

5.2 EMAIL NOTIFICATION TO RATERS

After test completion, an email is sent to all the raters who are registered with the test taker’s respective language. This requires identifying the correct raters based on the test taker’s language. The steps involved are:

- Find the language and testid of the test taker
- Get the registered raters from the database based on the test taker’s language

- Generate and send email to all these raters

When a test taker starts taking the test, the parameters related to that test taker are read from the servlet request object HttpServletRequest. When a test taker clicks on the submit button, a database query is executed using JDBC, passing the language being treated as a parameter. The existing database table (present in the CAST system) which contains information about the raters is “ADMINS” (Table 5.1) and the one which contains information about languages is “LANGS” (Table 5.2). Table structures for these tables are as follows:

Table 5.1. Table Structure for “ADMINS” Table

Name	Null?	Type
FNAME		VARCHAR2(60)
LNAME		VARCHAR2(60)
EMAIL		VARCHAR2(155)
ADDRESS		VARCHAR2(256)
CITY		VARCHAR2(256)
STATE		VARCHAR2(50)
COUNTRY		VARCHAR2(50)
ZIP		VARCHAR2(20)
PHONE		VARCHAR2(20)
LANG		VARCHAR2(50)
SUPERUSER		VARCHAR2(2)
USERID	NOT NULL	VARCHAR2(60)
PASSWD		RAW(50)
REGDATE		DATE
ACCESSLEVEL		VARCHAR2(50)
REF1		VARCHAR2(256)
REF1EMAIL		VARCHAR2(256)
REF2		VARCHAR2(256)
REF2EMAIL		VARCHAR2(256)
UA		VARCHAR2(1)

Table 5.2. Table Structure for "LANGS" Table

Name	Null?	Type
VALUE		VARCHAR2(25)
CODE	NOT NULL	VARCHAR2(6)

When the query is executed, it reads the raters' "EMAIL" from "ADMINS" table filtered by language and stores them in an array of strings. The CAST system already has a customized email sending functions developed using JavaMail API. The functions implement features specifically for Internet mails based on MIME standard. I used one of these functions which send email to all the raters from the array at the same time. The subject of the email is set to "CAST Test Completion Notification". The content of the email contains the text describing the test completion, the test taker's test id, and a link to the list of unrated tests page. Figure 5.1 shows the flowchart depicting the steps involved in sending emails to raters. Figure 5.2 shows the email sent to a rater on test completion.

5.3 LOCKING THE RATING PROCESS

Locking the rating process allows only one rater to rate a test at a time. To keep the locking process easier and manageable, I used a database table to keep track of the status of the rating process. The CAST system has a table named "BETAREG" (Table 5.3) which contains information about the tests (completed tests). Tests are identified by their testids. This table has following structure:

In "BETAREG" table the field named "RATED" stores a value to indicate whether the test has been rated or not. This field is not in use yet. So, I used it as a flag to indicate the status of the rating process. It can have following values:

- 'T' indicates the test has been rated
- 'F' indicates the test has not been rated
- 'H' indicates the test is currently being rated

Initially the "RATED" field is set to 'F' (when the test taker completes the test) for the record associated with the test taker's testid. When a rater opens his CAST account to view the list of unrated tests, it displays only those tests which have 'F' in their "RATED" field (Figure 5.3). Each test in the list has 'Rate This Test' link associated with it. It is a

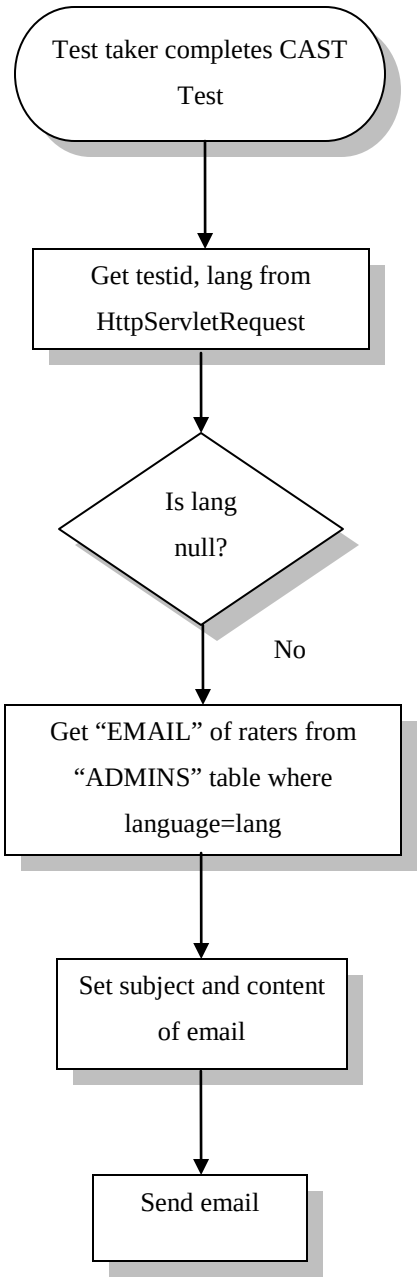

Test taker completes CAST Test
Get testid, lang from HttpServletRequest
Is lang null?
No
Get “EMAIL” of raters from “ADMINS” table where language=lang
Set subject and content of email
Send email

Figure 5.1. Flowchart depicting the steps involved in sending emails.

starting point of the rating process. The “RATED” field is set to ‘H’ when a rater clicks on ‘Rate This Test’ link for any test. This value remains to ‘H’ until the rater completes rating and submits the results or a situation occurs where the test has to be unlocked. When the test results are successfully submitted, this field is set to ‘T’. If the test is unlocked, this field is set to back to ‘F’.

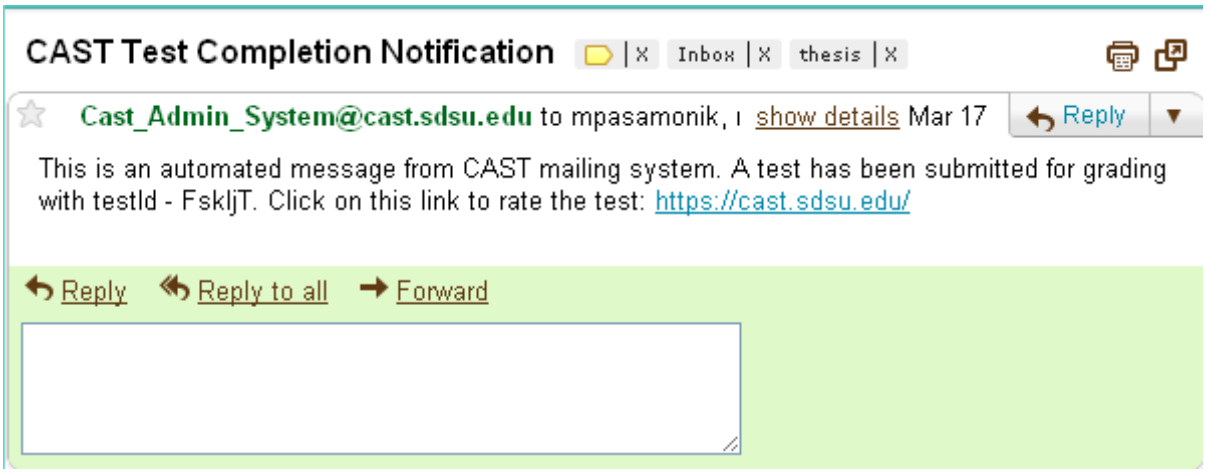

CAST Test Completion Notification
Inbox
thesis
Cast_Admin_System@cast.sdsu.edu to mpasamonik, i show details Mar 17
Reply
This is an automated message from CAST mailing system. A test has been submitted for grading with testId - FskljT. Click on this link to rate the test: https://cast.sdsu.edu/
Reply
Reply to all
Forward

Figure 5.2. Email sent on test completion.

Table 5.3. Table Structure for "BETAREG" Table

Name	Null?	Type
USERNAME		VARCHAR2(256)
ADDRESS		VARCHAR2(256)
CITY		VARCHAR2(100)
STATE		VARCHAR2(100)
ZIP		VARCHAR2(25)
USERID		VARCHAR2(256)
PASSEDTECH		VARCHAR2(4)
TESTID		VARCHAR2(10)
LANG		VARCHAR2(64)
RATED		VARCHAR2(2)
EMAIL	NOT NULL	VARCHAR2(256)
TESTDATE		DATE
ORGCODE		VARCHAR2(256)
TESTLEVEL		VARCHAR2(20)

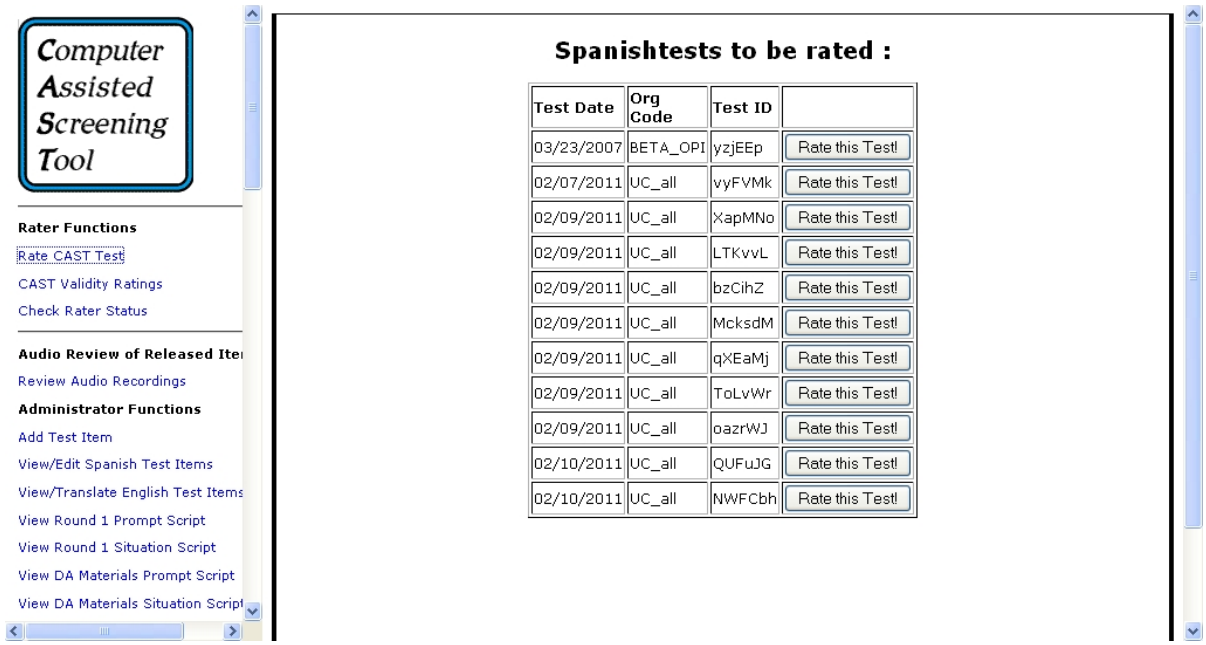

Computer Assisted Screening Tool
Rater Functions
Rate CAST Test
CAST Validity Ratings
Check Rater Status
Audio Review of Released Ite
Review Audio Recordings
Administrator Functions
Add Test Item
View/Edit Spanish Test Items
View/Translate English Test Items
View Round 1 Prompt Script
View Round 1 Situation Script
View DA Materials Prompt Script
View DA Materials Situation Script
Spanishtests to be rated :
Test Date
Org Code
Test ID
03/23/2007 BETA_OPI yzjEEp Rate this Test!
02/07/2011 UC_all vyFVMk Rate this Test!
02/09/2011 UC_all XapMNo Rate this Test!
02/09/2011 UC_all LTKvvL Rate this Test!
02/09/2011 UC_all bzCihZ Rate this Test!
02/09/2011 UC_all McksdM Rate this Test!
02/09/2011 UC_all qXEaMj Rate this Test!
02/09/2011 UC_all ToLvWr Rate this Test!
02/09/2011 UC_all oazrWJ Rate this Test!
02/10/2011 UC_all QUFuJG Rate this Test!
02/10/2011 UC_all NWFCbh Rate this Test!

Figure 5.3. List of tests viewed by the rater when he clicks on 'Rate CAST Test.'

When 'Rate This Test' link for a particular test is clicked, the "RATED" field in the "BETAREG" table is read for the testid associated with that test. The testid is read from the servlet request and is validated against the existing testids in the database before proceeding to the rating process. If it returns 'F', the rater is navigated to the rating page (Figure 5.4), and the "RATED" field is set to 'H'. If another rater tries to rate the same test, a popup will be displayed stating that the test is currently being rated by another rater (Figure 5.5). After that the test disappears from the list of unrated tests. Note that if a rater successfully starts rating a test, the unrated test list window disappears, so conflicts are not likely. When a rating has been submitted, the rating is replaced by the updated unrated test list window.

5.4 IMPLEMENTATION OF TIMER

The timer will count the time when rating process starts and notify the rater after certain amount of time. The requirements to set the timers are as follows:

- The timer should be set to 60 minutes
- The rater should be notified after 55 minutes of the rating process
- If the rater does not respond to the notification, the timer should expire after completing 60 minutes
- If the rater does respond, the timer should be incremented for another 60 minutes

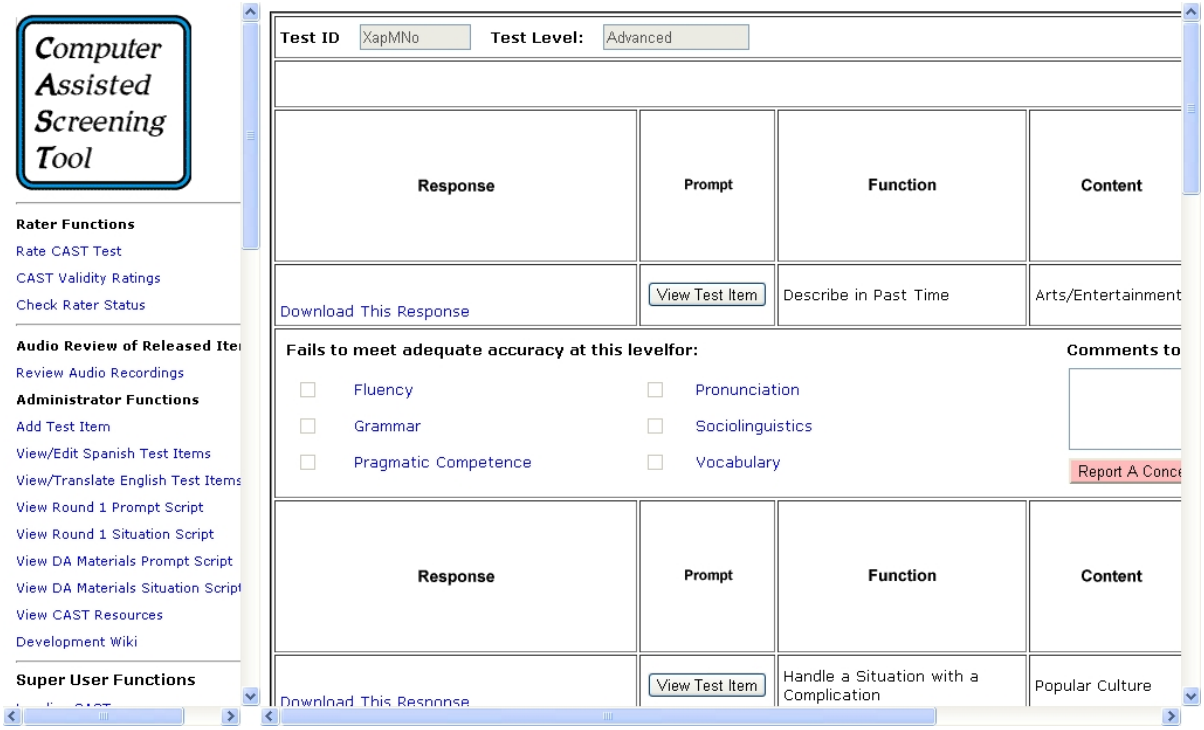


Figure 5.4. Rating page for testid 'XapMNo.'

Computer Assisted Screening Tool
Rater Functions
Rate CAST Test
CAST Validity Ratings
Check Rater Status
Audio Review of Released Ite
Review Audio Recordings
Administrator Functions
Add Test Item
View/Edit Spanish Test Items
View/Translate English Test Items
View Round 1 Prompt Script
View Round 1 Situation Script
View DA Materials Prompt Script
View DA Materials Situation Script
Spanishtests to be rated :
Test Date
Org Code
Test ID
02/07/2011 UC_all vyFVMk Rate this Test!
The page at http://130.191.246.127 says:
This test is currently being graded by other reviewer.
OK
02/09/2011 UC_all ToLvWr Rate this Test!
02/09/2011 UC_all oazrWJ Rate this Test!
02/10/2011 UC_all QUFuJG Rate this Test!
02/10/2011 UC_all NWFCbh Rate this Test!

Figure 5.5. Alert message when another rater tries to rate the same test.

To generate a timer, I used the “setTimeOut ()” method of JavaScript. The “setTimeOut ()” method allows you to execute a piece of JavaScript code in a specified number of milliseconds. Considering the above requirements, I created my own custom alert box [7]. It will be displayed to the rater after 55 minutes of the rating process. The reasons for choosing the custom alert box over JavaScript alert are:

- It allows code to execute in background
- It can be closed automatically
- The display of the alert box can be customized

The custom properties and JavaScript methods of custom alert box is defined in dialog_box.css and dialog_box.js respectively. The custom alert box consists of the following:

- A message which says ‘Your time is going to expire in 5 minutes. Press OK if you need extra time.
- Message type as ‘Warning’
- OK button

The custom alert box masks the background when displayed and waits for the user input. The page size including scrollbar is calculated before masking the background. If the rater clicks on “OK” button, the custom alert disappears and the rating process is resumed. If rater does not click on ”OK” button, the custom alert stays on screen for 5 minutes. After that it disappears automatically and the page is automatically navigated to rater home page (list of unrated tests).

The counter I created runs in background when the rating process is started. The function countTimeOut () contains the code to for implementing the timer. A variable is used to capture the button click. The countTimeOut () performs the following steps:

- Counter is set to 0 when the rating process starts and increments to 1000 every second.
- If the counter reaches 55000 milliseconds, a custom alert is displayed.
- If rater clicks on “OK” button, rating is resumed and the counter keeps incrementing. As the OK button is clicked, the user input is set to true. The default user input is false.
- When the counter reaches 60000 milliseconds, the user input is checked. If it is true, the counter is reset to 0. If it is false it navigates to the rater home screen and disappears automatically. The code to unlock the test is executed.

Figure 5.6 shows the flowchart depicting the steps involved in implementing the timer. Figure 5.7 shows the displayed custom alert message when the counter reaches 55000 milliseconds.

5.5 UNLOCKING

Once a test is locked, it will be unlocked only if the rater does not complete the rating process. A test is unlocked by executing the update query on the database table "BETAREG", setting the field "RATED" to 'F' for the testid of that test. To execute this update, the testid is required. To store the value of testid between client requests, I created a new session parameter "testId" to store the testid for which the rating process is taking place. There are two ways to remember parameters between pages:

1. Using Session
2. Using Cookie

The testid of the test selected for rating is stored in a session parameter, as the session can work even when cookies are disabled on the user's computer. Before setting testid to the session parameter "testId", the testid is validated to make sure that the test has been taken with that testid. The test is unlocked for the following cases:

- Case 1 - There is a session timeout:
  In CAST, the session expires after certain time of inactivity specified in setMaxInactiveInterval ( ). The setMaxInactiveInterval ( ) specifies the time, in number of seconds, between client requests before the servlet container will invalidate the session [8]. To execute update on database on session timeout, I created a new servlet "sessionListener.java". This servlet implements the HttpSessionListener Interface. Implementations of this interface are notified of changes to the list of active sessions in a web application [9]. To receive notification events, the implementation class must be configured in the deployment descriptor for the web application. This interface contains "sessionCreated ()" and "sessionDestroy ()" methods. The entry of the servlet "sessionListener.java" is made in the deployment descriptor of web.xml file. When the session is created, "sessionCreated ()" method is called and when the session is about to be invalidated, the "sessionDestroy ()" method is called. I did override "sessionDestroy ()" method and added code to execute update on "BETAREG" table upon a session timeout.
- Case 2 - Rater logs out or navigates away from rating page without submitting the test results:
  If the rater clicks on any other link of CAST site while rating the test or opens another webpage in the same browser tab, he will be navigated away from the

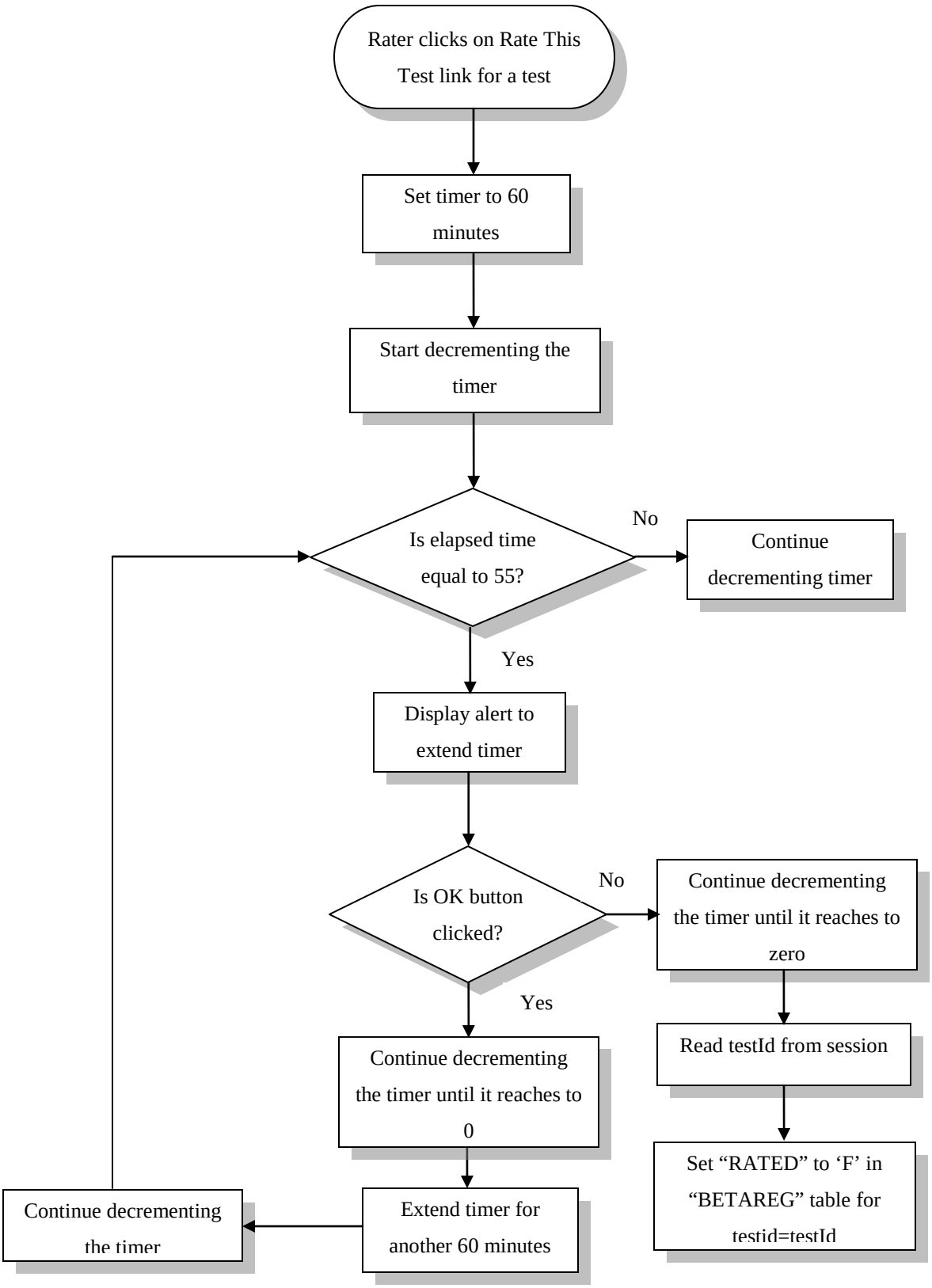

Rater clicks on Rate This Test link for a test
Set timer to 60 minutes
Start decrementing the timer
Is elapsed time equal to 55?
No
Continue decrementing timer
Yes
Display alert to extend timer
Is OK button clicked?
No
Continue decrementing the timer until it reaches to zero
Yes
Read testId from session
Continue decrementing the timer until it reaches to 0
Set "RATED" to 'F' in "BETAREG" table for testid=testId
Extend timer for another 60 minutes
Continue decrementing the timer

Figure 5.6. Flowchart depicting the steps involved in executing the timer.

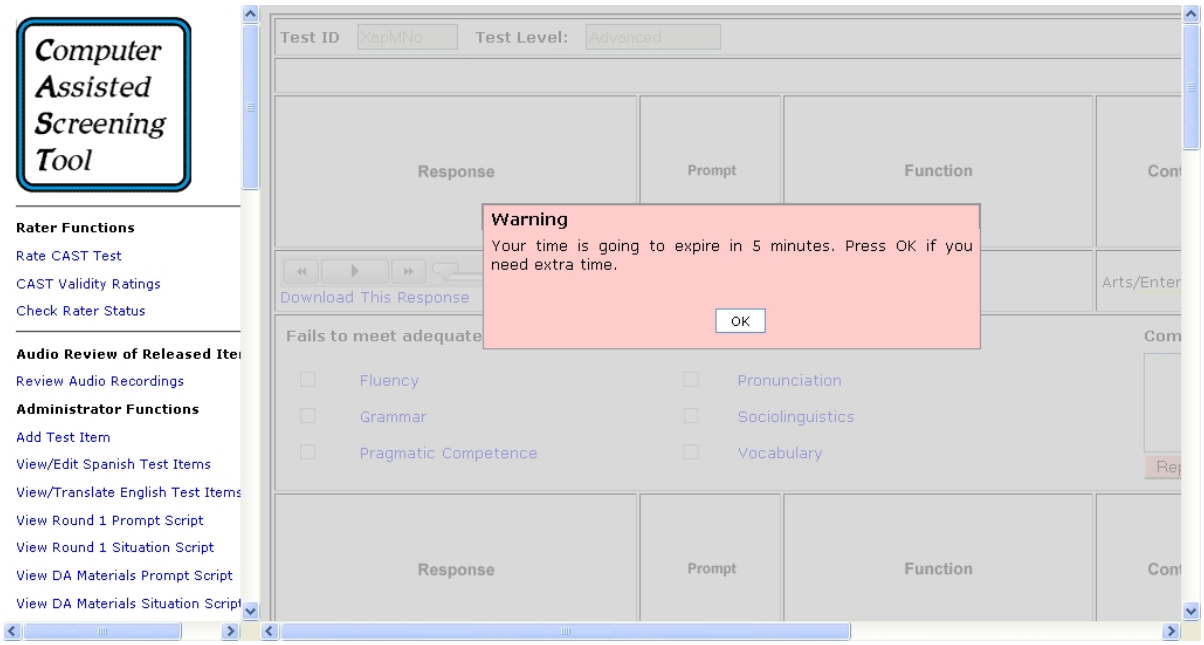


Figure 5.7. The custom alert message to notify the rater about the timer.

rating process. To execute database update after navigation I created a method "updateTestAvailability ()". This method is created in base class of CAST system. It takes testid as a parameter and contains the code to execute update on "BETAREG" table for that testid. Once the code to update the database is executed, the "testId" is set to null in the session variable to prevent repeated execution of the same code on further navigation. This method is called on logout and navigation and it will be executed if "testId" in session contains some value. In CAST, pages are cached and these cached copies are updated every certain interval of time. The problem arises when navigation between rating and other pages happens too quickly and cached copies of pages are not updated. In this case, these cached pages will not reflect the current session parameters. And hence, the test will not be unlocked. To overcome this problem, the servlet for rating is reloaded every time, when the page is navigated back to rating. Caching is disabled by setting parameter "Cache-Control" to "no-store" in response object header [10].

- Case 3 – Back button is clicked:
  The functionality to enable unlocking on back button is similar to the functionality for navigation. While clicking on back button, a fresh copy of the rating servlet is loaded. So it reads the updated "testId" from the session and updates "BETAREG" table for that testid (sets "RATED" to 'F' on back button). It sets it back to 'H' if forward button is clicked.

- Case 4 - Browser tab or window is closed:
  One of the most difficult tasks in this project is to capture browser close event on various browsers. The best way to capture browser closing is to know when the page is being unloaded [11]. The page is unloaded when

1. The user navigates to different URL
2. Attempts to close the browser/tab window
3. The page is reloaded or refreshed

The JavaScript “onunload ()” and “onbeforeunload ()” events fire when the document is unloaded. These methods are called from HTML body tag and specify the JavaScript code to execute when a page unload occurs. My solution works for Internet Explorer and Firefox. For Internet Explorer, “onunload” event is not recognized and in some of the Firefox versions, “onbeforeunload” event is not recognized. So, I used “onbeforeunload ()” method for Internet Explorer and “onunload ()” for Firefox [12]. The header and body for the resulting webpage are generated in CASTUtil.java and are common for all the webpages generated in CAST. So the code needed to be executed only when the browser/tab window of the rating page is closed, and not when any other page is unloaded as a result of other actions. In the solution, I used a new session parameter “page”. This parameter stores the name of the page. When the rating process starts, I set this parameter to “rating”, and I set it to null when the rating results are submitted. When onunload or onbeforeunload events are fired and the “page” parameter stores “rating”, it indicated that rating page is being unloaded.

Another major issue was to execute java code (server side code) from onbeforeunload and onunload (client side events). When browser is closed, a new request to the server needed to be sent to execute java code on the server side. To allow this, I created one empty html form with just id and action attributes. The id identifies the form and the action attribute specifies where to send the form data when a form is submitted. I created one new servlet “updateDatabase.java” which reads the “testId” from a session and if it is not null and if it is valid (the test has been taken with that testid), it executes update on “BETAREG” for that testid. In my form, the action attribute specifies the path to this “updateDatabase.java” servlet. When onunload or onbeforeunload events are fired, this form is submitted. It submits request to “updateDatabase.java” servlet and this way the database update is executed on browser close. For all other browsers, when browser/tab window of rating page is closed, the database update is executed when the session is destroyed on timeout. The session destroys automatically after predefined time of inactivity. So when the browser is closed the session is destroyed after the time specified in setMaxInactiveInterval ().

- Case 5 – Rater does not respond to alert message of timer:
  If rater is prompted to extend the timer for rating process, and the rater does not respond, the timer expires after 5 minutes. After the timer expires, the servlet “rate.java” is reloaded to display the list of unrated tests. The database update is automatically executed when the servlet is reloaded. The test is unlocked and the list of unrated tests is displayed.

Figure 5.8 shows the flowchart depicting the steps involved in implementing locking and unlocking mechanism.

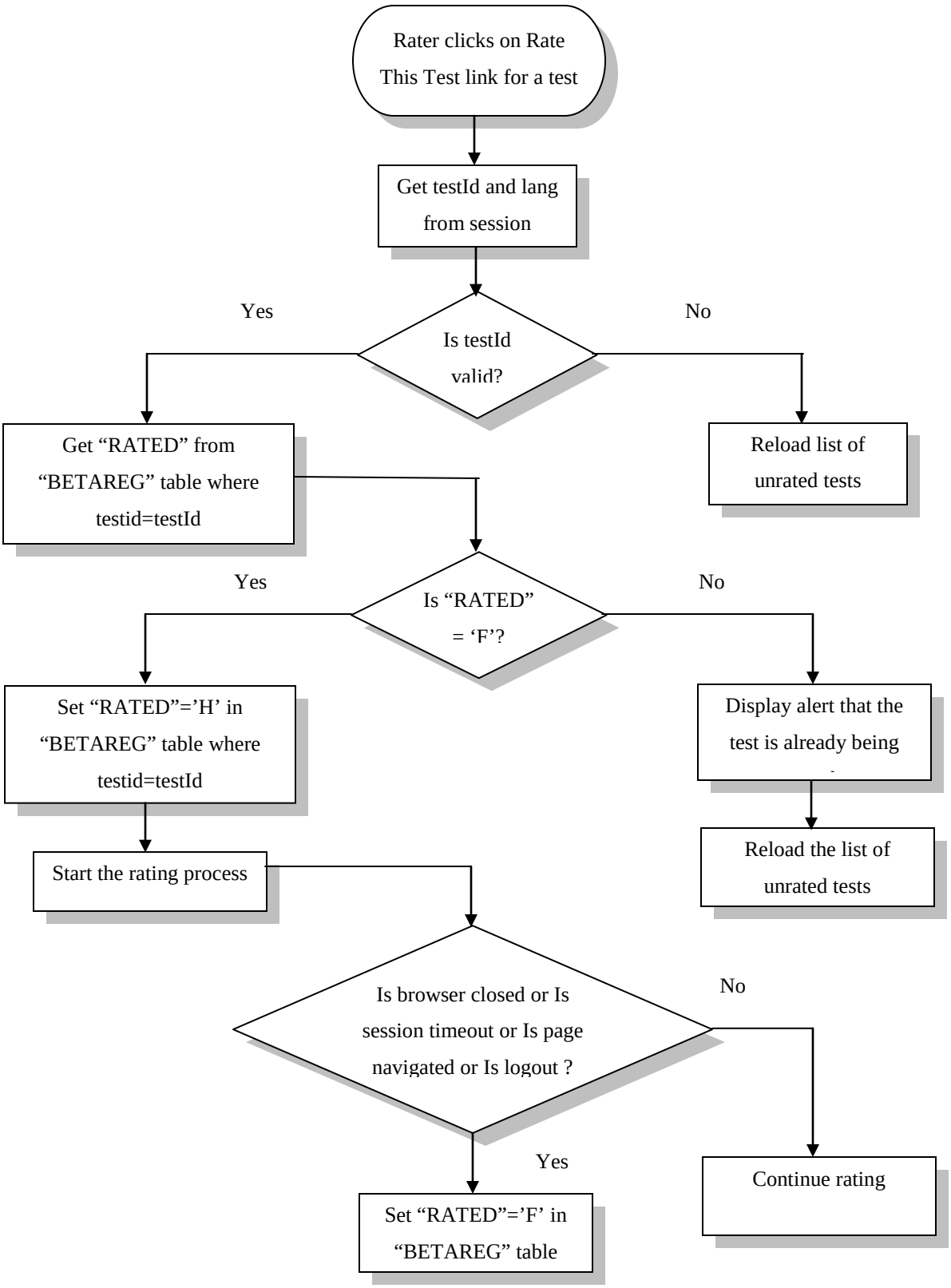

Rater clicks on Rate This Test link for a test
Get testId and lang from session
Is testId valid?
Yes
No
Get "RATED" from "BETAREG" table where testid=testId
Reload list of unrated tests
Is "RATED" = 'F'?
Yes
No
Set "RATED"='H' in "BETAREG" table where testid=testId
Display alert that the test is already being
Reload the list of unrated tests
Start the rating process
Is browser closed or Is session timeout or Is page navigated or Is logout ?
No
Yes
Set "RATED"='F' in "BETAREG" table
Continue rating

Figure 5.8. Flowchart depicting the locking and unlocking process.

5.6 ISSUES WITH LOCKING AND UNLOCKING

Several issues required attention while developing this project. One of them was to coordinate my code with the existing functionality of the CAST system. There were a number of situations where the locking mechanism was not behaving as expected. The problems and the solutions to these problems are as follows:

1. In the existing functionality, the same session variable is shared within the same browser window. I used a session variable to store the testid and so all other tabs of the same browser window read the same "testId" from the session. When the rater navigates away from the rating process or closes the browser, the database update was executed using the "testId" from the session. So clicking on any link of CAST site from another tab or closing any other tab of the same browser window was unlocking the test. The rater was able to evaluate the same test simultaneously from another tab of same browser window (Figure 5.9). Also, displaying the rating page and displaying the list of unrated page are performed by the same servlet. To coordinate with these functionalities, I created two more session variables "PageName" and "CurrPage". The "PageName" parameter is used to check whether the test is being rated or not and "CurrPage" parameter is used to check whether the current page displays a list of unrated tests. While the rating is in process, "PageName" carries the value "rate". While rater opens list of unrated tests, "CurrPage" carries the value "testList". At the beginning of any unlocking code, these two parameters are checked. The unlocking code is executed only when the "PageName" carries "rate" and "CurrPage" is not "testList". So when the rater will try to rate the same test in another tab, it will display an alert message (Figure 5.10) This solution is implemented for Internet Explorer and Firefox. For Safari and Chrome, the rater will be warned at the beginning of the rating process not to open multiple tabs of CAST site in the same browser (Figure 5.11). To know the current browser, I created a method called "getBrowserName". This method gets the browser name from request header and returns the browser name and its version.
2. The entire rating page is considered as a form and this form is submitted to another servlet for further processing. I also created a form to submit on page unload. Submitting these two forms together was producing an unexpected behavior. To overcome this problem, I changed the existing rating form declaration to submit it explicitly instead of submitting it on "onSubmit" method of HTML forms. When the button is clicked to submit the form, a JavaScript function was called to perform validations. I added the form as an argument to pass it to this function and this function submits the form explicitly by executing form.submit. Also, Internet Explorer was giving problem in recognizing execution sequence for these two forms. Also, UI in Internet explorer was changed. To deal with this, I declared both the forms at the same place one by one to overcome. The form with test rating should be submitted first and then the form for onunload should get submitted.

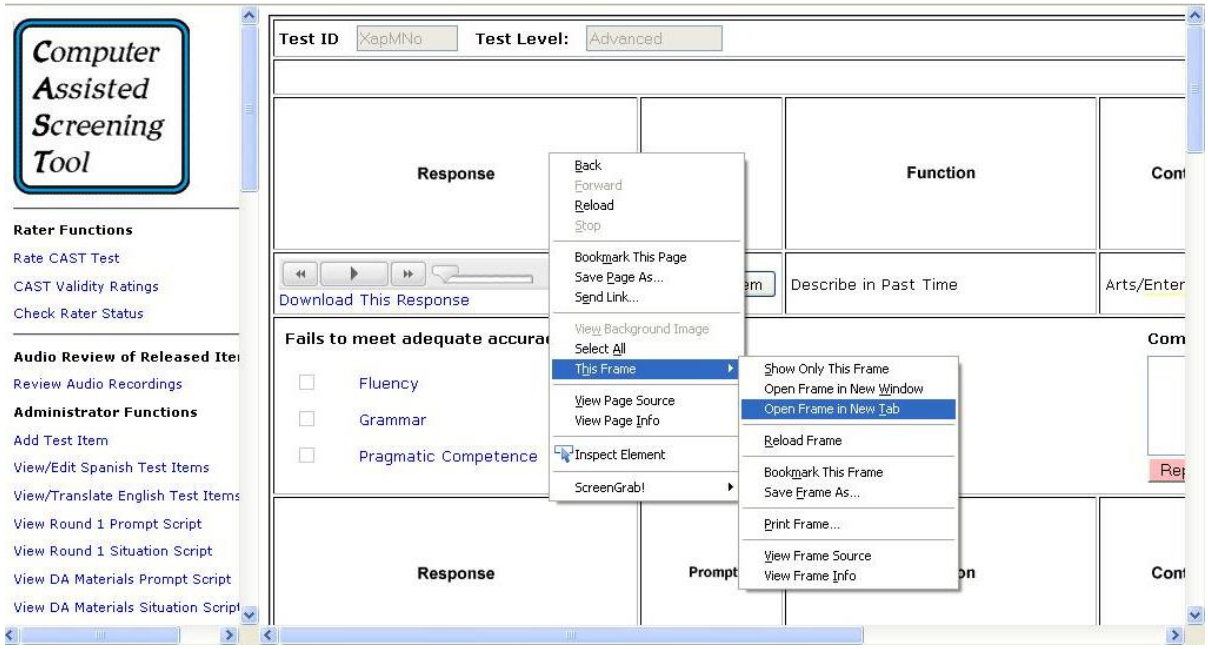


Figure 5.9. A rater trying to rate the same test on another tab.

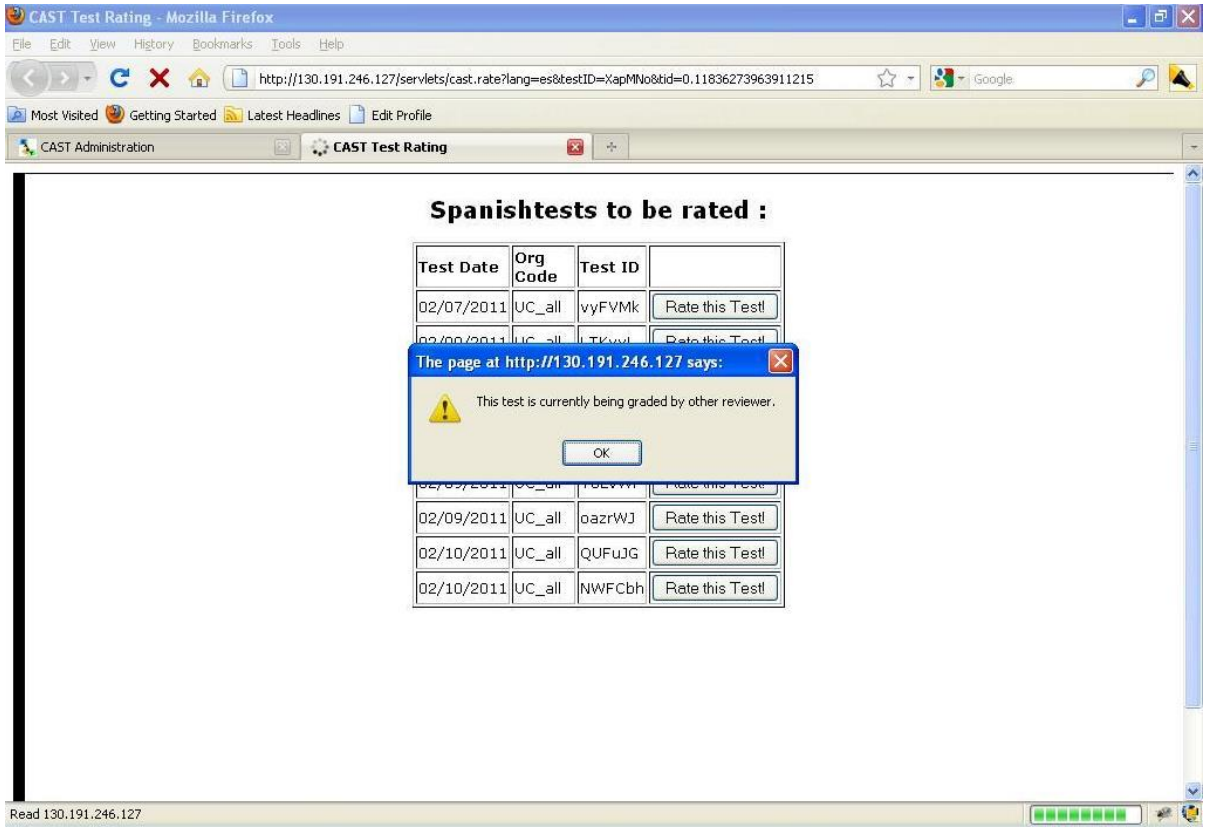

CAST Test Rating - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://130.191.246.127/servlets/cast.rate?lang=es&testID=XapMNo&tid=0.11836273963911215
Most Visited Getting Started Latest Headlines Edit Profile
CAST Administration
CAST Test Rating
Spanishtests to be rated :
Test Date
Org Code
Test ID
02/07/2011 UC_all vyFVMk Rate this Test!
The page at http://130.191.246.127 says:
This test is currently being graded by other reviewer.
OK
02/09/2011 UC_all oazrWJ Rate this Test!
02/10/2011 UC_all QUFuJG Rate this Test!
02/10/2011 UC_all NWFCbh Rate this Test!
Read 130.191.246.127

Figure 5.10. Alert displayed when a rater tries to rate the same test on another tab.

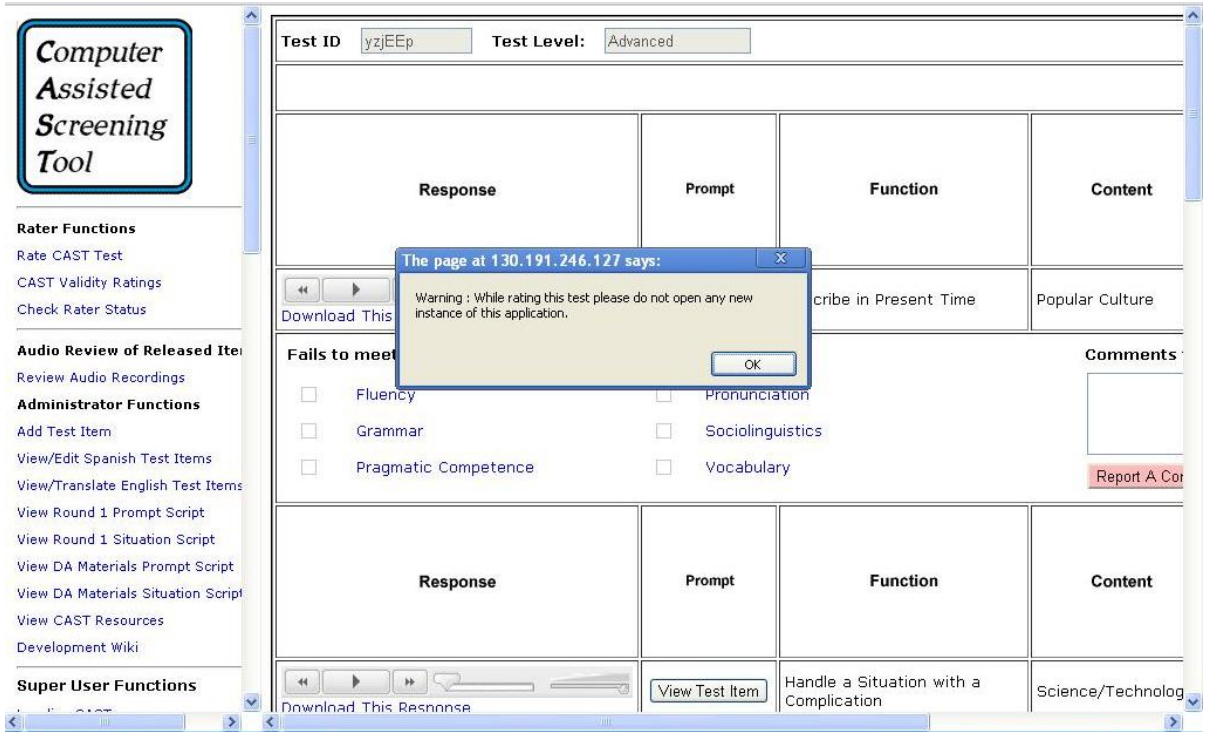


Figure 5.11. Warning on Chrome when rating starts.

CHAPTER 6

SECURITY

In this project, the data is stored and retrieved from the database by performing several SQL operations. Also, the data is passed between pages using GET and POST methods. The security of this approach is ensured by preventing SQL injection and by performing back-end validations of the data.

6.1 SQL INJECTION OVERVIEW

This section discusses about SQL injection and the ways to prevent it. SQL Injection is a code injection technique that exploits security vulnerability in some computer software [13]. It is a method by which the parameters of a Web-based application are modified in order to change the SQL statements that are passed to a database to return data [14]. An attack against a database using SQL Injection could be motivated by two primary objectives:

1. To steal data from a database from which the data should not normally be available, or to obtain system configuration data that would allow an attack profile to be built.
2. To gain access to an organization's host computers via the machine hosting the database.

There are four main categories of SQL Injection attacks against databases [15]:

1. SQL Manipulation: Manipulation is process of modifying the SQL statements by using various operations. One way for implementing SQL Injection using SQL Manipulation method is by changing the where clause of the SQL statement to get different results.
2. Code Injection: Code injection is process of inserting new SQL statements or database commands into the vulnerable SQL statement. One of the possible code injection attacks is to append EXECUTE command to the vulnerable SQL statement.
3. Function Call Injection: Function call injection is process of inserting various database function calls into a vulnerable SQL statement. This function calls can be made to manipulate data into the database.
4. Buffer Overflows: Buffer overflow is caused by using function call injection.

Oracle, like any other database product is vulnerable to SQL injection attacks. The following abuses can be inflicted on an Oracle or other databases:

- UNIONS can be added to an existing statement to execute a second statement
- SUBSELECTS can be added to existing statements

- Existing SQL can be short-circuited to bring back all data
- Data Definition Language (DDL) can be injected
- INSERTS, UPDATES and DELETES can also be injected
- Other databases can be injected through the first by using database links
- A large selection of installed packages and procedures are available to read and write O/S files

6.2 SQL INJECTION PREVENTION TECHNIQUES

As long as injected SQL code is syntactically correct, it is hard to detect tampering programmatically. Programs designed to run in a secure environment can be copied to a non-secure environment [16]. There are some actions that can be taken in order to prevent SQL injection. All the user input should be validated and the code should be carefully reviewed that executed SQL commands on the server in use. The following things should be considered in order to prevent SQL injection:

1. Escaping User Input: The user input should be escaped before putting into a query. If you escape the entire user supplied input using the proper escaping scheme for the database you are using, the DBMS will not confuse that input with SQL code written by the developer, thus avoiding any possible SQL injection vulnerabilities.
2. Use of PreparedStatement: JBDC PreparedStatements are simple to write, and easier to understand than dynamic queries. PreparedStatement uses parameterized queries. Parameterized queries force the developer to first define all the SQL code, and then pass in each parameter to the query later. This coding style allows the database to distinguish between code and data, regardless of what user input is supplied. Prepared statements ensure that an attacker is not able to change the intent of a query, even if SQL commands are inserted by an attacker.
3. Stored Procedures: The stored procedure should use input validation or proper escaping to make sure that all user supplied input to the stored procedure is not being used to inject SQL code into the dynamically generated query. Stored procedures require the developer to define the SQL code first, and then pass in the parameters after. Stored procedures have the same effect as the PreparedStatement. The difference between PreparedStatement and stored procedures is that the SQL code for a stored procedure is defined and stored in the database itself, and then called from the application.
4. Use of minimum privileges: The privileges assigned to the database account in the environment should be minimized in order to minimize the potential damage of a successful SQL injection attack. DBA or admin type access rights should only be assigned to people working closely with the application. Also, the privileges of the operating system account that the DBMS runs under should be minimized.

5. White List Input Validation: Input validation can be used to detect unauthorized input before it is passed to the SQL query. White list validation is appropriate for all the fields provided to the server to perform database operations. White list validation involves defining exactly what is authorized and what is not authorized.

6.3 TECHNIQUES USED IN THIS PROJECT

To prevent SQL injection I used the following techniques in my code:

1. Escaping User Input: For this I created one “sanitizeInput_select” method. This method replaces all special characters and key words (words that have some meaning in database) with space. The only session parameter I passed in database queries is “testId”. So, this method is called from all the places where “testId” is going to be used in a query, after reading it from the current session.
2. Use of PreparedStatement: I replaced all the JDBC query statements in all the servlets with PreparedStatement.
3. White List Input Validation: To check the whether the testid passed to the server is one of the existing testids, I created a method “validTestId”. This method checks the testid against the existing testids into the database “BETAREG” table. The testid is first sanitized and then is checked against the maximum allowed length (varchar (10)). If testid passes through these filters, it is then checked against the existing testids in the database by using SELECT query to read the number of rows matching this testid. If it returns only one row then the testid is the valid testid.

CHAPTER 7

FUTURE ENHANCEMENTS

The solutions I provided are compatible with Firefox, Internet Explorer, Safari and Chrome. Some of the functionalities are only implemented for Firefox and Internet Explorer. These are:

- Detecting browser/tab close event from rating page
- Supporting multiple tabs of the same application while rater is rating a test

The solutions can be extended to implement on Safari and Chrome to improve the performance of CAST on these browsers. Moreover, the rating process notifications can be localized to specific language of the rater which will make it easier to understand for the raters. The CAST test is currently computer based, which can be developed into a mobile application to improve the learning process.

REFERENCES

[1] *CAST homepage*. CAST, http://cast.sdsu.edu, accessed August 2010, n.d.

[2] *LARC homepage*. LARC, http://larc.sdsu.edu, accessed August 2010, n.d.

[3] *Java servlet*. Wikipedia, http://en.wikipedia.org/wiki/Java_Servlet, accessed September 2010, n.d.

[4] *What is a servlet?* Sun Microsystems, http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Servlets2.html#75087, accessed September 2010, n.d.

[5] *Oracle database*. Wikipedia, http://en.wikipedia.org/wiki/Oracle_Database, accessed September 2010, n.d.

[6] *ACID*. Wikipedia, http://en.wikipedia.org/wiki/ACID, accessed September 2010, n.d.

[7] HELBRAX, *Custom alert boxes using JavaScript and the DOM*. The Code Project, http://www.codeproject.com/KB/scripting/DOMAlert.aspx, accessed October 2010, n.d.

[8] *Interface HttpSession*. Oracle, http://download.oracle.com/javaee/1.3/api/javax/servlet/http/HttpSession.html, accessed September 2010, n.d.

[9] *Interface HttpSessionListener*. Oracle, http://download.oracle.com/javaee/1.4/api/javax/servlet/http/HttpSessionListener.html, accessed September 2010, n.d.

[10] J. HUNTER, *Servlet best practices, part 2*. O'Reilly on Java, http://onjava.com/pub/a/onjava/excerpt/jebp_3/index2.html, accessed October 2010, n.d.

[11] *JavaScript events*. W3schools, http://www.w3schools.com/js/js_events.asp, accessed October 2010, n.d.

[12] AAMIR HASAN, *Difference between JavaScript OnUnload VS OnBeforeUnload events*. aspx Tutorial, http://www.aspxtutorial.com/post/2011/02/06/Difference-between-javascript-OnUnload-VS-OnBeforeUnload-Events.aspx, accessed October 2010, n.d.

[13] *SQL injection*. Wikipedia, http://en.wikipedia.org/wiki/SQL_injection, accessed January 2011, n.d.

[14] PETE FINNIGAN, *SQL injection and Oracle, part one*. Symantec, http://www.symantec.com/connect/articles/sql-injection-and-oracle-part-one, accessed January 2011, n.d.

[15] SAGAR JOSHI, *SQL injection attack and defense*. Packet Source, http://www.packetsource.com/article/sql-injection/40060/sql-injection-attack-and-defense, accessed January 2011, n.d.

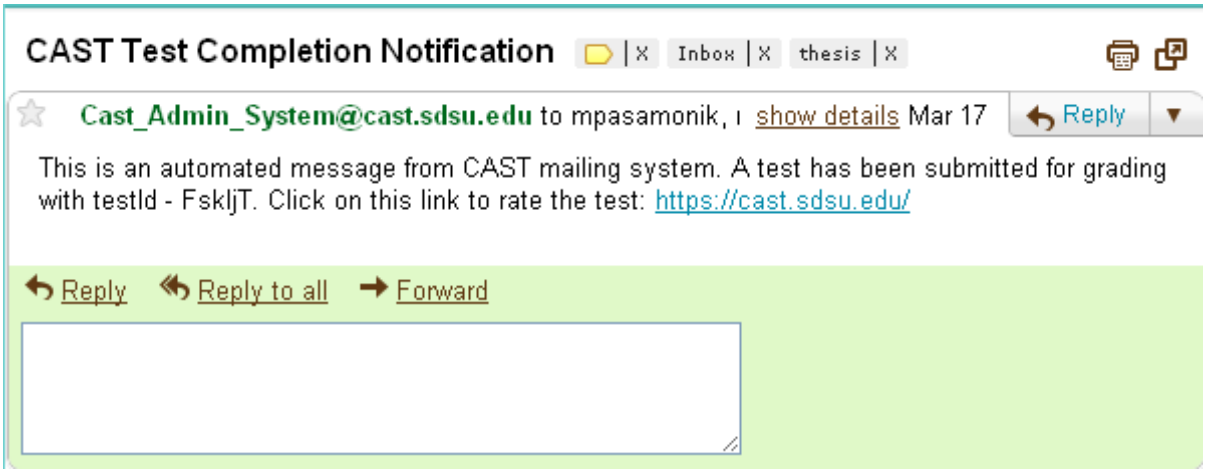
[16] *SQL injection*. Microsoft Developer Network [MSDN], http://msdn.microsoft.com/en-us/library/ms161953.aspx, accessed January 2011, n.d.

APPENDIX

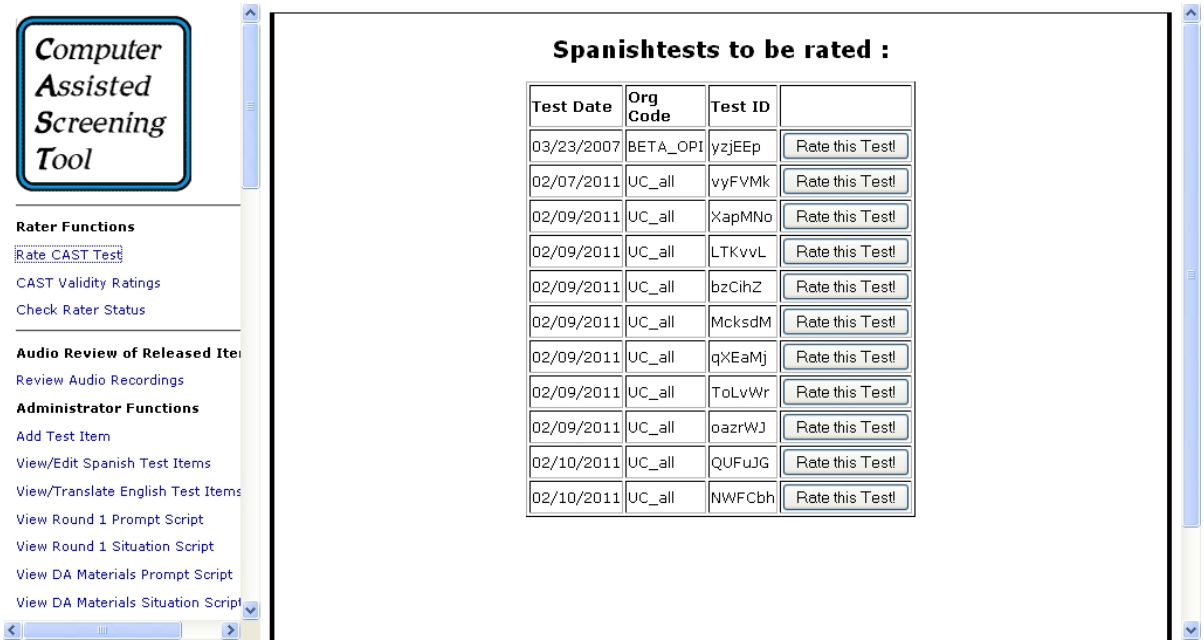
USER MANUAL

This instruction explains the functionalities added to the rating process of CAST. It is designed as an instruction guide for the raters who are the language expert and evaluate the CAST tests taken in their target language.

1. The raters will receive a notification by email if a test in their registered language is submitted for grading. The email will consist of the testid of the test taker and a link to the CAST website.


CAST Test Completion Notification
Inbox
thesis
Cast_Admin_System@cast.sdsu.edu to mpasamonik, i show details Mar 17
Reply
This is an automated message from CAST mailing system. A test has been submitted for grading with testId - FskljT. Click on this link to rate the test: https://cast.sdsu.edu/
Reply
Reply to all
Forward

2. When 'Rate CAST Test' link is clicked, the only tests which have not been graded yet or which are not currently being graded by other raters, will be loaded.


Computer Assisted Screening Tool
Rater Functions
Rate CAST Test
CAST Validity Ratings
Check Rater Status
Audio Review of Released Ite
Review Audio Recordings
Administrator Functions
Add Test Item
View/Edit Spanish Test Items
View/Translate English Test Items
View Round 1 Prompt Script
View Round 1 Situation Script
View DA Materials Prompt Script
View DA Materials Situation Script
Spanishtests to be rated :
Test Date
Org Code
Test ID
03/23/2007 BETA_OPI yzjEEp Rate this Test!
02/07/2011 UC_all vyFVMk Rate this Test!
02/09/2011 UC_all XapMNo Rate this Test!
02/09/2011 UC_all LTKvvL Rate this Test!
02/09/2011 UC_all bzCihZ Rate this Test!
02/09/2011 UC_all McksdM Rate this Test!
02/09/2011 UC_all qXEaMj Rate this Test!
02/09/2011 UC_all ToLvWr Rate this Test!
02/09/2011 UC_all oazrWJ Rate this Test!
02/10/2011 UC_all QUFuJG Rate this Test!
02/10/2011 UC_all NWFCbh Rate this Test!

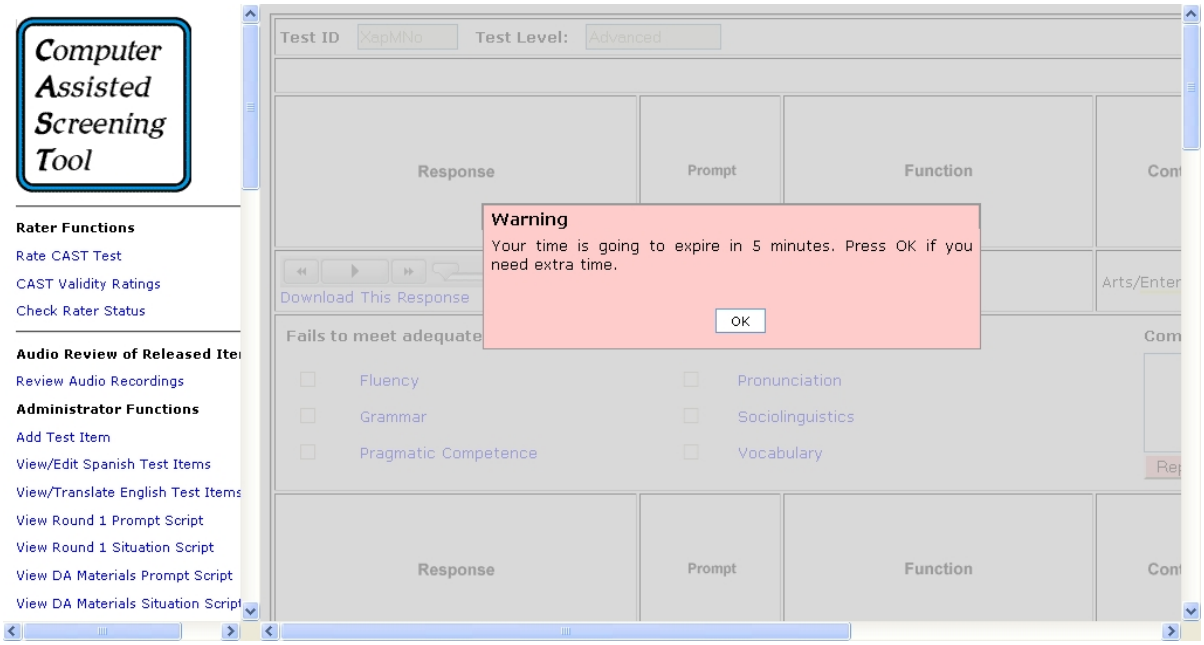
After the rating starts, the test will be locked for that rater and will disappear from the list of unrated tests. If a rater has already started rating a test and another rater tries to rate the same test, an alert message will be displayed, and the test will disappear from the list of unrated tests.

Computer Assisted Screening Tool
Rater Functions
Rate CAST Test
CAST Validity Ratings
Check Rater Status
Audio Review of Released Ite
Review Audio Recordings
Administrator Functions
Add Test Item
View/Edit Spanish Test Items
View/Translate English Test Items
View Round 1 Prompt Script
View Round 1 Situation Script
View DA Materials Prompt Script
View DA Materials Situation Script
Spanishtests to be rated :
Test Date
Org Code
Test ID
02/07/2011 UC_all vyFVMk Rate this Test!
02/09/2011 UC_all ToLvWr Rate this Test!
02/09/2011 UC_all oazrWJ Rate this Test!
02/10/2011 UC_all QUFuJG Rate this Test!
02/10/2011 UC_all NWFCbh Rate this Test!
The page at http://130.191.246.127 says:
This test is currently being graded by other reviewer.
OK

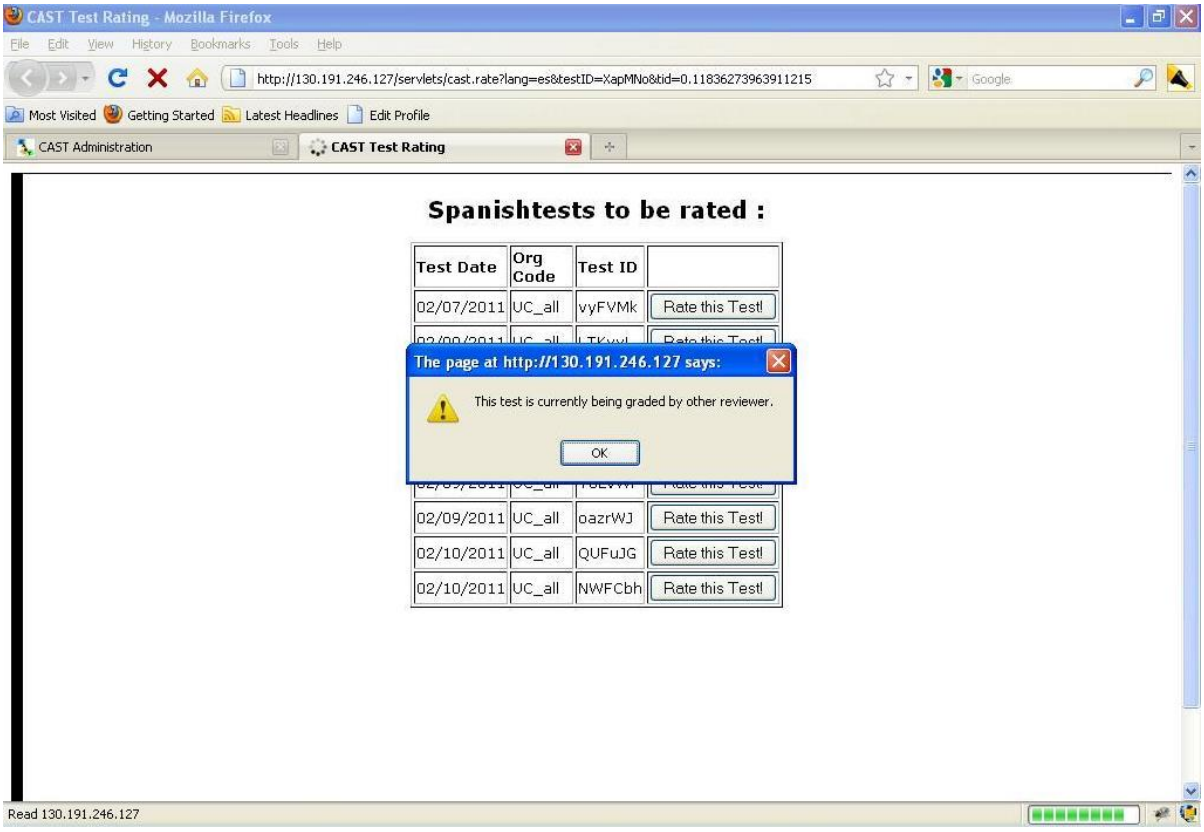
The locked test will be unlocked if the rater fails to finish the rating process. The test will be unlocked for the following scenarios:

- When there is a session timeout
- When the rater navigates away from the rating page
- When the rater logs out
- When the rater closes the browser/tab window
- When the rater clicks on back button

3. A time slot of 60 minutes is allocated to the rating process. A rater can finish the test evaluation in 60 minutes. However, after 55 minutes of rating, a notification message will be displayed to the rater about the timer expiration. The rater can extend the timer by responding to the notification. The timer will expire automatically after 5 minutes if the rater does not respond to the notification and the rater will be navigated away from the rating screen. The test locked for the rater, will then be unlocked.



4. In Firefox and Internet Explorer browsers, the raters will not be able to open the same rating page on two different tabs of the same browser window. An alert message will be displayed when a rater tries to do the same.



If a rater is using Chrome or Safari browsers, he cannot open multiple instance of CAST website within the same browser window while he is rating. A warning message will be displayed when the raters starts rating a test.